

**DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING**

**EEE416(18/19) Coding and Cryptography**

*Lab 2*

*EEE408*

**Lab Report**

|  |  |  |
| --- | --- | --- |
| Student Name | : | Enge Xu |
| Student ID | : | 1821635 |
| Date | : | 2019/5/25 |
| Professor | : | Jiming Xiao |

# ABSTRACT

This assessment aims at evaluating students’ ability to exploit the deep learning knowledge, which is accumulated during lectures, and after-class study, to analyze, design, implement, develop, test and document the pedestrian detection algorithm using Fast R-CNN framework. The assessment will be based on the Pytorch software.

**Overall Description**

This lab is to use the Pytorch software and Fast R-CNN framework for pedestrian detection.edestrian detection aims to predict bounding boxes of all the pedestrian instances in an image. It as attracted much attention within the computer vision community in recent years as an important omponent for many human-centric applications, such as self-driving vehicles, person re . identification, video surveillance and robotics.

Fast R-CNN is a fast framework for object detection using deep convolutional neural network. The ollowing images are examples of pedestrian detection.

**TASK**

**Question**

1. Please describe the 2 key components in the Fast R-CNN framework: the RolPooling layer and the loss functions in the framework. (20%)
2. Please describe the object detection performance metric, mAP (Mean Average Precision), and explain why it can well reflect the object detection accuracy. (20%).
3. Please train and test the Fast R-CNN framework on one of the existing pedestrian detection datasets, and report the final AP performance that you have achieved. The dataset could be CUHK-SYSU [21, Citypersons [3]. Please also report some pedestrian detection examples by including the images and bounding boxes. (40%).
4. Propose your own method to further improve the pedestrian detection performance based on.the Fast R-CNN framework. (20%)

Hint:

1. In literature, there are existing methods that improve the Fast R-CNN for pedestrian detection.One such work is published in [4]. You can refer to [4] for problem 4.

1. There are many open sourced codes for Faster R-CNN and it is highly recommended to use the latest benchmark from Facebook (https://eithub.com/facebookresearch/maskrcnn-benchmark).and train the model without Mask part.
2. Python version of the API of the datasets can be found on github. And the datasets can be download athttps://pan.baidu.com/s/15ZknOJOMBva9JmwHM2ykbw mm: n6se <https://www.cityscapes-dataset.com/>

**Answer**

**Question 1**

A network layer, which can be regarded as a single-layer sppnet, is called ROI Pooling. This network layer can map input of different sizes to a fixed scale feature vector.As we all known, conv, pooling, relu operations such as don't need the input of fixed size. Therefore, in the original image after performing these operations, although the input image size is different to get the feature of the map size is different also, not directly receive a full connection layer, but you can join this fantastic ROI pooling layer. For each region, said a fixed dimension feature extracting by normal is used to identify the type of the softmax again.

The idea is as follows:

The region proposal is divided into a grid of H \* W size.

MaxPooling (i.e., one output value for each grid) is done for each grid.

Combine all output values to form a feature map of fixed size H \* W.

Multiple loss fusion (classification loss and regression loss fusion). Log loss (i.e., take negative log of the probability of true classification and output K+1 dimension) is adopted for classification. The loss of regression is basically the same as r-cnn.

The total loss function is as follows:

L(p,u,tu,v)=Lcls(p,u)+λ[u⩾1]Lloc(tu,v)

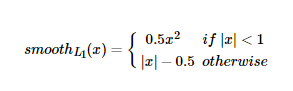
The classification loss function is as follows:

Lcls(p,u)=−log pu

The regression loss function is as follows:

Lloc(tu,v)=∑iϵ{x,y,w,h}smoothL1(tui−vi)

It includes:



**Question 2**

Several important parameters of performance evaluation in the target detection model include accuracy, precision and recall rate.We will discuss a common metric: mean average precision, or MAP.

In binary classification, accuracy and recall rate are simple and intuitive statistics. However, there is a difference in target detection: even if our object detector detects an object in the image, it is useless if we cannot find which position in the image it is in.Since we need to predict the occurrence and location of the target in the image, the calculation accuracy and recall rate are different from the ordinary dichotomies.

Reasons for choosing MAP: The classification and localization of models in the target detection problem need to be evaluated, and each image may have different targets of different categories.Therefore, the standard measurement used in the image classification problem cannot be directly applied to the target detection problem.

Preicision is what percentage of a positive sample you think is actually a positive sample, and recall is what percentage of a positive sample you find.

Core question: we need a threshold for score. Why?In a bounding box, for example, I identified the duck has the highest score, but he also only 0.1, then he is really a duck?Probably he's still a negative sample.So we need a threshold. If the duck is recognized and the score is greater than this threshold, we can really say that it is a positive sample, otherwise it is a negative sample.

So how does threshold affect precision and recall exactly?We have used the example of ducks. If the threshold is too high and the prediction is strict, we believe that all ducks are basically ducks, and the precision is high.But also because of too strict screening, we also spared some score lower ducks, so the recall is lower if the threshold is too low, what will be as a duck, its precision is very low, recall will be very high.

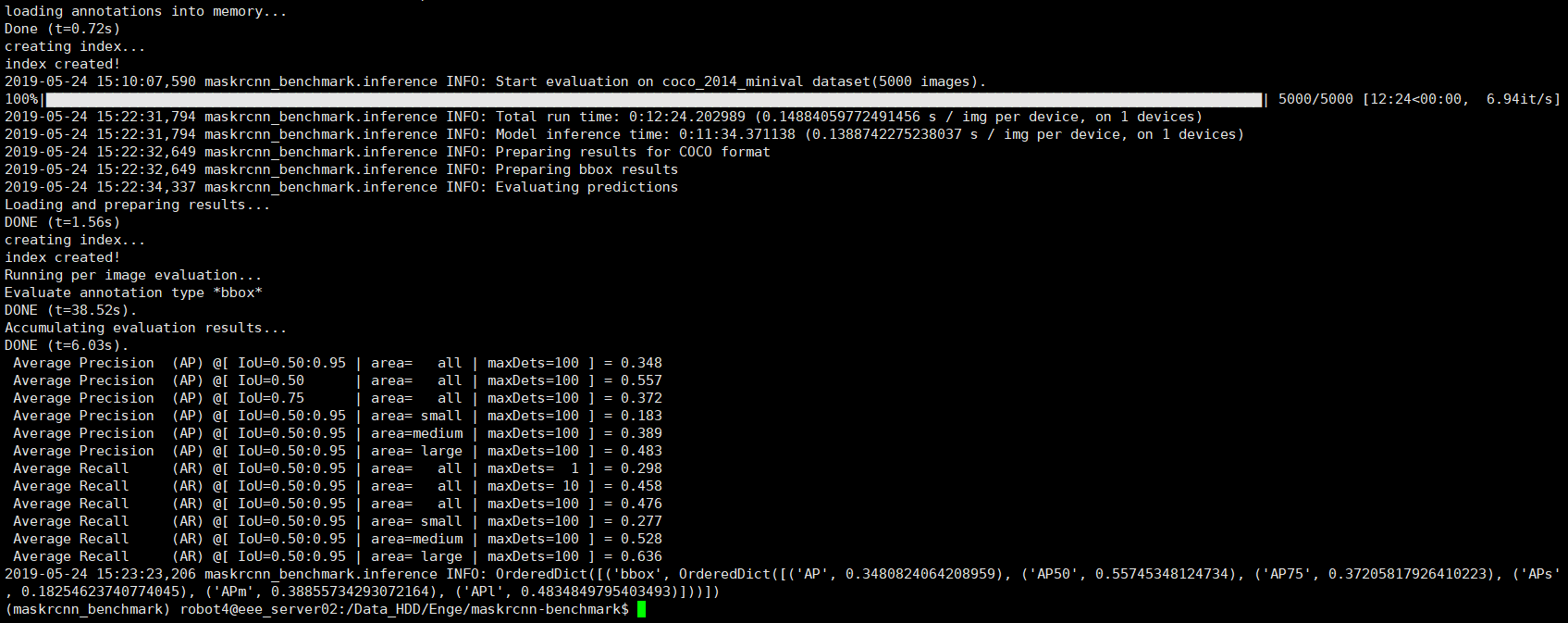
So we can clear the threshold does influence the precision and recall of the ducks and the trend of change, also has brought the thinking, precision is not an absolute thing, but the relative threshold and change the contents,Recall similarly, it is not appropriate to use precision as the standard for single judgment.This is a trade off between precision and recall, which is not fully expressed by a set of fixed values, because we can get different (or possibly the same) precision recall values according to different thresholds.

In this way, for each threshold, we have a pair of precision and recall, so there is the curve relationship between precision and recall. There is such a precise-recall curve, which measures the relationship between two valuable judgment standards, precision and recall. So, it is better to consider them dynamically together and get the Average precision of the duck class.That is, the area under curve, which can fully represent the overall advantages and disadvantages of precision and recall in this model.

Finally, we calculate the Average Precision of each class, and we get the mean Average Precision.

**Question 3**

After downloading the program from github, it failed seven or eight more times before finally getting something to test. The task itself is not complicated, but it needs enough time to trial and error. The final result of running the provided test program with the trained model is shown as below (The model is under the file of Data\_HDD/Enge/maskrcnn-benchmark);



**Question 4**

Due to the lack of time, the model can not be modified with code. But Some suggestions may be useful in practical application.

Each mini-batch in the fine-tuning of the author consists of 2 images and 64 ROIs collected from each image.In addition, during training, each Image will be flipped horizontally with a probability of 0.5 to enhance the data.No data enhancement was used beyond that. Therefore, some more data enhancement can be added into the code, like zoom, crop, pan, and noise. But the actual function may be questioned because the number of samples is sufficient.

The problem with Fast r-cnn: there is a bottleneck: selective search, finding all the candidate boxes, this is also very time-consuming.Can we find a more efficient way to find these candidate boxes?

Solution: add a neural network to extract edges, and the job of finding candidate boxes is left to the neural network.

Specific practices:

• put RPN behind the last convolutional layer

• RPN direct training to get candidate areas

This is an improvement made by faster RCNN.